Exploratory Data Analysis

The HOUSES dataset contains a collection of recent real estate listings in San Luis Obispo county and around it. The dataset is as a CSV file. The dataset contains the following fields:

1. MLS: Multiple listing service number for the house (unique ID).
2. Location: city/town where the house is located. Most locations are in San Luis Obispo county and northern Santa Barbara county (Santa Maria-Orcutt, Lompoc, Guadelupe, Los Alamos), but there some out of area locations as well.
3. Price: the most recent listing price of the house (in dollars).
4. Bedrooms: number of bedrooms.
5. Bathrooms: number of bathrooms.
6. Size: size of the house in square feet.
7. Price/SQ.ft: price of the house per square foot.
8. Status: type of sale. Thee types are represented in the dataset: Short Sale, Foreclosure and Regular.

Lets import the required libraries that we will be using later.

```
In [1]: from numpy import * # everything
        import pandas as pd
```

Let's load the dataset into a pandas dataframe and have a look at the headers.

```
In [2]: df = pd.read_csv('data.csv', sep=',', error_bad_lines=False) # read fie
         as a dataframe
```

Lets take a look at the first 2 rows of the dataframe.

```
In [3]: df.head(2)
```

Out[3]:

	MLS	Location	Price	Bedrooms	Bathrooms	Size	Price/SQ.Ft	Status
0	132842	Arroyo Grande	795000	3	3	2371	335.30	Short Sale
1	134364	Paso Robles	399000	4	3	2818	141.59	Short Sale

Examine the provided columns, does the pandas infered datatype of each column make sense? Include your code and/or comments below.

In [4]:

```
#TODO
print (df["MLS"])
print (df["Location"])
print(df["Price"])
print (df["Bedrooms"])
print (df["Bathrooms"])
print (df["Size"])
#The only  datatype that does not make sense is Location, location shoul
d have string as its datatype.
```

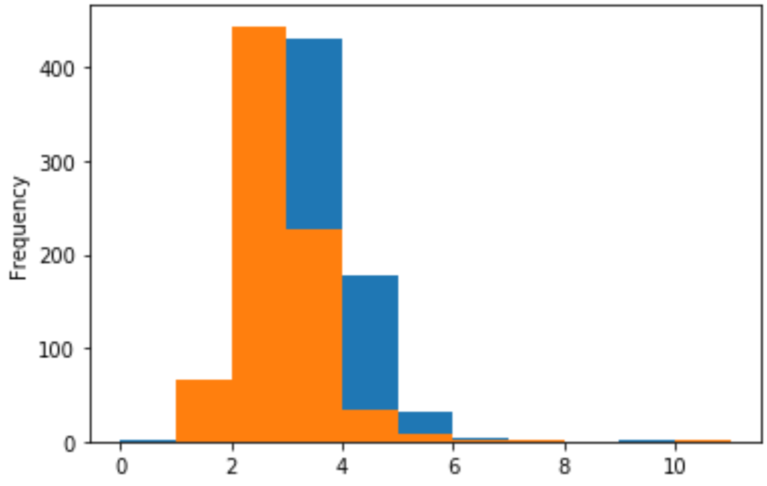
```
0      132842
1      134364
2      135141
3      135712
4      136282
        ...
776    154562
777    154565
778    154566
779    154575
780    154580
Name: MLS, Length: 781, dtype: int64
0           Arroyo Grande
1             Paso Robles
2             Paso Robles
3               Morro Bay
4      Santa Maria-Orcutt
              ...
776           Paso Robles
777           Paso Robles
778       San Luis Obispo
779         Arroyo Grande
780               Cambria
Name: Location, Length: 781, dtype: object
0       795000
1       399000
2       545000
3       909000
4       109900
        ...
776     319900
777     495000
778     372000
779     589000
780    1100000
Name: Price, Length: 781, dtype: int64
0      3
1      4
2      4
3      4
4      3
      ..
776    3
777    3
778    3
779    3
780    3
Name: Bedrooms, Length: 781, dtype: int64
0      3
1      3
2      3
3      4
4      1
      ..
776    3
777    2
```

```
778    2
779    2
780    3
Name: Bathrooms, Length: 781, dtype: int64
0      2371
1      2818
2      3032
3      3540
4      1249
       ...
776    1605
777    1877
778    1104
779    1975
780    2392
Name: Size, Length: 781, dtype: int64
```

Next, lets look at a specific column or feature in the dataframe. Based on the provided dataset, what are the distinct number of bedrooms and bathrooms? Hint : Use the unique function https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.unique.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.unique.html)

```
In [5]: # TODO
        #import numpy as np
        #import pandas as pd
        #pd.DataFrame
        df["Bedrooms"].plot.hist()
        df["Bathrooms"].plot.hist()
        #pandas.unique(Bedrooms)
```

```
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7f2b8bbdcf28>
```



What if we want to drop a column from the dataframe, like the 'Location' column. Hint: Use the drop function https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.drop.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.drop.html)

In [6]:

```
# TODO
df.drop(columns=['Location'])
```

Out[6]:

	MLS	Price	Bedrooms	Bathrooms	Size	Price/SQ.Ft	Status
0	132842	795000	3	3	2371	335.30	Short Sale
1	134364	399000	4	3	2818	141.59	Short Sale
2	135141	545000	4	3	3032	179.75	Short Sale
3	135712	909000	4	4	3540	256.78	Short Sale
4	136282	109900	3	1	1249	87.99	Short Sale
...	...	...	...	...	...	...	...
776	154562	319900	3	3	1605	199.31	Regular
777	154565	495000	3	2	1877	263.72	Regular
778	154566	372000	3	2	1104	336.96	Foreclosure
779	154575	589000	3	2	1975	298.23	Regular
780	154580	1100000	3	3	2392	459.87	Regular

781 rows × 7 columns

Let's rename the first column.

Hint: A Google search for 'python pandas dataframe rename' points you at this documentation https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.rename.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.rename.html)

In [7]:

```
print ("Before rename", df.columns)
#TODO
df = df.rename(columns={"MLS": "mls"})
df = df.rename(columns={"Location": "loc"})
df = df.rename(columns={"Price": "prc", "Bedrooms": "bedrs", "Bathrooms"
: "bath"})
print ("After rename", df.columns)
```

```
Before rename Index(['MLS', 'Location', 'Price', 'Bedrooms', 'Bathroom
s', 'Size',
       'Price/SQ.Ft', 'Status'],
      dtype='object')
After rename Index(['mls', 'loc', 'prc', 'bedrs', 'bath', 'Size', 'Pric
e/SQ.Ft', 'Status'], dtype='object')
```

What is the max, min, mean/avg, and standard deviation of the column 'Bedrooms'?

In [8]:

```
# TODO
print (' Max: ', df.bedrs.max())
print (' Min: ', df.bedrs.min())
print ('Mean: ', df.bedrs.mean())
print ('  SD: ', df.bedrs.std())
```

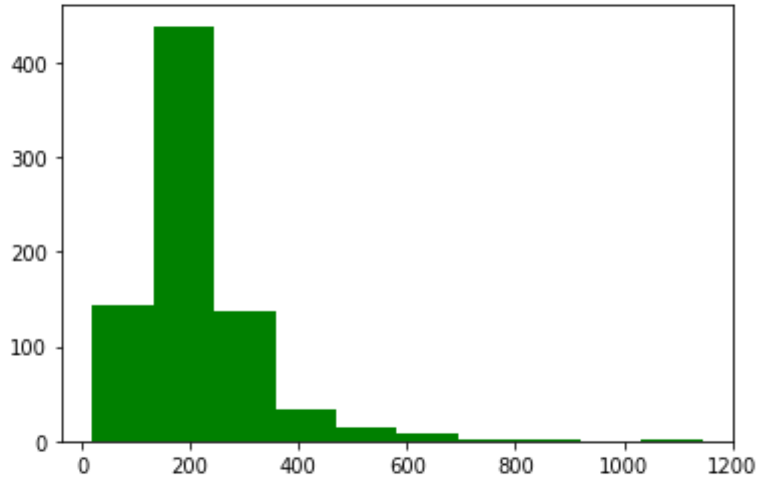
```
 Max:  10
 Min:  0
Mean:  3.1421254801536493
  SD:  0.8557678151609314
```

Plot the distribution of 'Price/SQ.Ft' using matplotlib

In [9]:

```
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt

# plot histogram
n, bins, patches = plt.hist(df['Price/SQ.Ft'], 10, facecolor='green')
plt.show()
```



One of the best ways to inspect data is visualize it. One way to do this is by using a scatter plot. A scatter plot of the data puts one feature along the x-axis and another along the y-axis, and draws a dot for each data point.

Since its difficult to visualize more than 2 or 3 features, one possibility is to use a pair plot that looks at all possible pairs of features. The pair plot shows the interaction of each pair of features inorder to visualize any correlation between features.

In [10]:

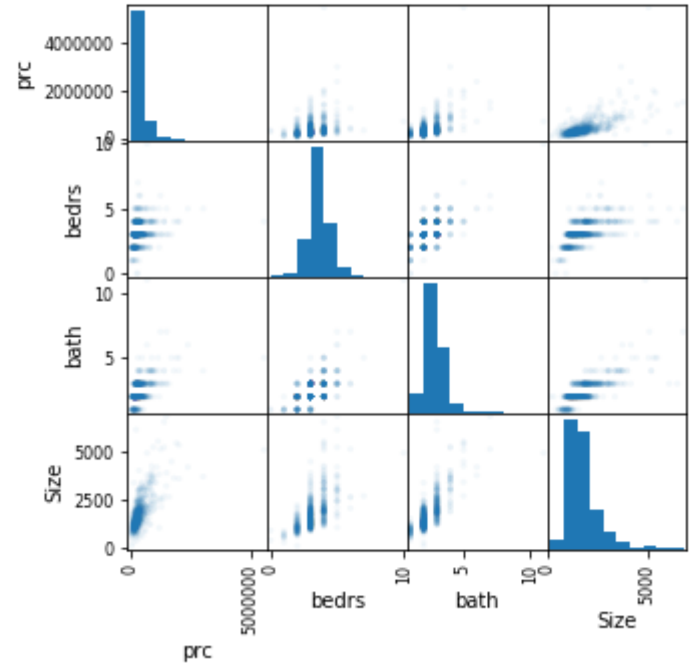
```
# import the scatter_matrix functionality
import random as rand
import numpy as np
import pandas as pd
pd.DataFrame
from pandas.plotting import scatter_matrix

import matplotlib.pyplot as plt

print (df.shape)
x = df.iloc[:,[1,2,3,4,5]] # extract only a subset of columns from dataf
rame (using index)
y = x.dropna(thresh=5) # drop any rows that have 5 or more fields as NAN
#a = pd.scatter_matrix(x, alpha=0.05, figsize=(5,5), diagonal='hist')
a = scatter_matrix(x, alpha=0.05, figsize=(5,5), diagonal='hist')

plt.show()
```

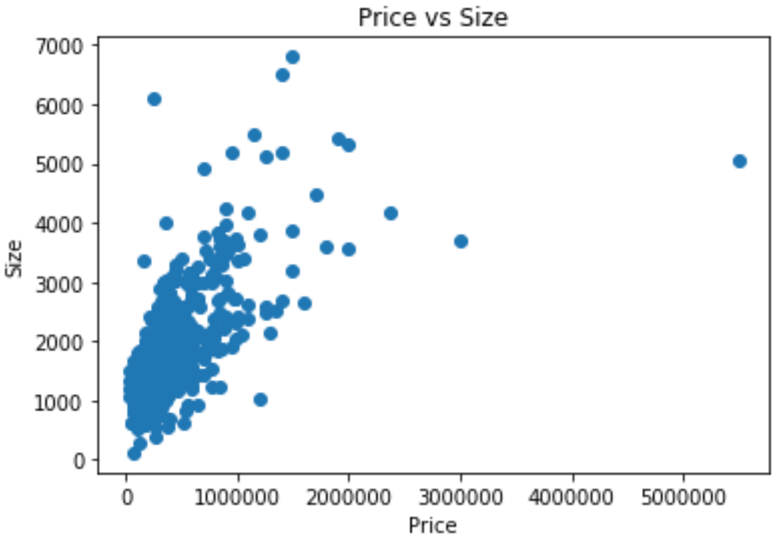
```
(781, 8)
```



In [11]:

```
#Lets plot the Price vs Size of the homes

fig=plt.figure()
plt.scatter(df.prc, df.Size)
axis = fig.gca() #get current axis
axis.set_title('Price vs Size')
axis.set_xlabel('Price')
axis.set_ylabel('Size')
fig.canvas.draw()
```



What does the visualizations and the statistics we observed tell you so far. Is there any other interesting stats or visualizations you think might be helpful. Include your comments and code below

# TODO These visualitzations show us that there is a pretty linear correlation when price and size are small, however once size reaches the value around 3000 there is not as much correlation between the two.

Categorical Encoding

If we have categorical or continuous variables and we would like to encode them into discrete integer files (like 0, 1, 2, ...) we can use several tricks in pandas to do this.

In [12]:

```
# Approach 1 - Pandas makes it easy for us to directly replace the
# text values with their numeric equivalent by using replace .


newValues = {"Status": {"Foreclosure": 1, "Short Sale": 2, "Regular" : 3
}}
df2 = df.replace(newValues, inplace=False )
df2.head()
```

Out[12]:

	mls	loc	prc	bedrs	bath	Size	Price/SQ.Ft	Status
0	132842	Arroyo Grande	795000	3	3	2371	335.30	2
1	134364	Paso Robles	399000	4	3	2818	141.59	2
2	135141	Paso Robles	545000	4	3	3032	179.75	2
3	135712	Morro Bay	909000	4	4	3540	256.78	2
4	136282	Santa Maria-Orcutt	109900	3	1	1249	87.99	2

In [13]:

```
# Approach 2 - Another approach to encoding categorical values is to use
a technique called label encoding.
# Label encoding is simply converting each value in a column to a numbe
r.

# One trick you can use in pandas is to convert a column to a category,
 then use those category
# values for your label encoding.

df["Status"] = df["Status"].astype('category')
df.dtypes

# Then you can assign the encoded variable to a new column using the ca
t.codes accessor.
df["Status_cat"] = df["Status"].cat.codes
df.head()
```

Out[13]:

	mls	loc	prc	bedrs	bath	Size	Price/SQ.Ft	Status	Status_cat
0	132842	Arroyo Grande	795000	3	3	2371	335.30	Short Sale	2
1	134364	Paso Robles	399000	4	3	2818	141.59	Short Sale	2
2	135141	Paso Robles	545000	4	3	3032	179.75	Short Sale	2
3	135712	Morro Bay	909000	4	4	3540	256.78	Short Sale	2
4	136282	Santa Maria-Orcutt	109900	3	1	1249	87.99	Short Sale	2

In [14]:

```
"""Approach 3 - Label encoding has the advantage that it is straightforw
ard but it has the
   disadvantage that the numeric values can be “misinterpreted” by the a
lgorithms. For example,
   the value of 1 is obviously less than the value of 3 but does that re
ally correspond to the data set in real life?
   For example, is "Foreclosure" =1 closer to "Short Sale" =2 compared t
o "Regular" =3?

   A common alternative approach is called one hot encoding. The basic s
trategy is to convert each category value
   into a new column and assigns a 1 or 0 (True/False) value to the colu
mn. This has the benefit of not weighting
   a value improperly but does have the downside of adding more columns
 to the data set.

   Pandas supports this feature using get_dummies. This function is name
d this way because it creates
   dummy/indicator variables (aka 1 or 0)."""

pd.get_dummies(df, columns=["Status"], prefix=["new"]).head()

# basically, it creates a 3 new columns (one for each unique value in th
e column.) with the prefix "new_"
```

Out[14]:

	mls	loc	prc	bedrs	bath	Size	Price/SQ.Ft	Status_cat	new_Foreclosure	new_R
0	132842	Arroyo Grande	795000	3	3	2371	335.30	2	0	
1	134364	Paso Robles	399000	4	3	2818	141.59	2	0	
2	135141	Paso Robles	545000	4	3	3032	179.75	2	0	
3	135712	Morro Bay	909000	4	4	3540	256.78	2	0	
4	136282	Santa Maria-Orcutt	109900	3	1	1249	87.99	2	0	

Submission Instructions

Once you are finished, follow these steps:

Restart the kernel and re-run this notebook from beginning to end by going to Kernel > Restart Kernel and Run All Cells.

If this process stops halfway through, that means there was an error. Correct the error and repeat Step 1 until the notebook runs from beginning to end. Double check that there is a number next to each code cell and that these numbers are in order.

Then, submit your lab as follows:

Go to File > Export Notebook As > PDF.

Double check that the entire notebook, from beginning to end, is in this PDF file. (If the notebook is cut off, try first exporting the notebook to HTML and printing to PDF.)

Upload the PDF to iLearn.

Have the TA check your lab to obtain credit.